upper bounds for cod[x] and dom[x]

Johan G. F. Belinfante
2009 January 1

```
In[1]:= SetDirectory["l:"]; << goedel.08dec31a;<< tools.m

        :Package Title: goedel.08dec31a         2008 December 31 at 11:50 p.m.

        It is now:  2009 Jan 1 at 11:3

        Loading Simplification Rules

        TOOLS.M                                      Revised 2008 December 26

        weightlimit = 40
```

summary

Various upper bounds for **cod[x]** and **dom[x]** are derived. No restrictions are placed on the class **x**, although the results are expected to be mainly of interest for categories. Counterexamples are provided to show that each of the inclusions derived is in general strict, even for monoids.

dom

Theorem.

```
In[2]:= SubstTest[subclass, composite[id[v], u], u, {u → domain[x], v → ids[x]}] // Reverse
Out[2]= subclass[dom[x], domain[x]] == True

In[3]:= subclass[dom[x_], domain[x_]] := True
```

cod

Theorem.

```
In[4]:= SubstTest[subclass, composite[u, id[v]], u, {u → domain[x], v → ids[x]}] // Reverse
Out[4]= subclass[inverse[cod[x]], domain[x]] == True

In[5]:= subclass[inverse[cod[x_]], domain[x_]] := True
```

neutrality and divisibility

For every ternary relation **x** there are binary left and right divisibility relations **composite[x, inverse[FIRST]]** and **composite[x, inverse[SECOND]]**, respectively . Explicitly, if **member[pair[pair[u, v], w], x]**, let us say that **u** is a **left-divisor** of **w** and **v** is a **right-divisor** of **w**. If **member[pair[pair[u, v], v], x]** let us say that **u** is **left-neutral** for **v** and simililarly if **member[pair[pair[u, v], u], x]**, let us say that **v** is right-neutral for **u**. The **left-neutrality relation** is

```
In[6]:= class[pair[u, v], member[pair[pair[u, v], v], x]]
```

```
Out[6]= fix[composite[inverse[SECOND], x]]
```

Lemma.

```
In[7]:= Map[implies[and[member[u, y], member[v, z]], #] &,
          SubstTest[implies, and[member[pair[u, t], composite[Id, z]],
            member[pair[t, w], composite[Id, y]]], member[pair[u, w], composite[y, z]],
           {t → pair[u, v], y → x, w → v, z -> inverse[FIRST]}]] // Reverse
```

```
Out[7]= or[member[pair[u, v], composite[x, inverse[FIRST]]], not[member[u, y]],
          not[member[v, z]], not[member[pair[pair[u, v], v], x]]] == True
```

```
In[8]:= or[member[pair[u_, v_], composite[x_, inverse[FIRST]]],
          not[member[pair[pair[u_, v_], v_], x_]],
          not[member[u_, y_]], not[member[v_, z_]]] := True
```

Theorem. The left-neutrality relation is a subclass of the left-divisibility relation.

```
In[9]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class,
          pair[u, v], or[member[pair[u, v], t], not[member[u, V]], not[member[v, V]],
           not[member[pair[pair[u, v], v], x]]], t -> composite[x, inverse[FIRST]]]]
```

```
Out[9]= subclass[fix[composite[inverse[SECOND], x]], composite[x, inverse[FIRST]]] == True
```

```
In[10]:= subclass[fix[composite[inverse[SECOND], x_]], composite[x_, inverse[FIRST]]] := True
```

Lemma.

```
In[11]:= Map[implies[and[member[u, y], member[v, z]], #] &,
           SubstTest[implies, and[member[pair[u, t], composite[Id, z]],
             member[pair[t, w], composite[Id, y]]], member[pair[u, w], composite[y, z]],
            {t → pair[v, u], w → v, y → x, z -> inverse[SECOND]}]] // Reverse
```

```
Out[11]= or[member[pair[u, v], composite[x, inverse[SECOND]]], not[member[u, y]],
           not[member[v, z]], not[member[pair[pair[v, u], v], x]]] == True
```

```
In[12]:= or[member[pair[u_, v_], composite[x_, inverse[SECOND]]],
           not[member[pair[pair[v_, u_], v_], x_]],
           not[member[u_, y_]], not[member[v_, z_]]] := True
```

Corollary. A similarly inclusion for right-neutrality and right-divisibilty.

```
In[13]:= SubstTest[subclass, fix[composite[inverse[SECOND], t]],
           composite[t, inverse[FIRST]], t → flip[x]] // Reverse

Out[13]= subclass[inverse[fix[composite[inverse[FIRST], x]]],
           composite[x, inverse[SECOND]]] == True

In[14]:= subclass[inverse[fix[composite[inverse[FIRST], x_]]],
           composite[x_, inverse[SECOND]]] := True
```

Corollary.

```
In[82]:= SubstTest[subclass, inverse[u], inverse[v],
           {u -> inverse[fix[composite[inverse[FIRST], x]]],
            v -> composite[x, inverse[SECOND]]}] // Reverse

Out[82]= subclass[fix[composite[inverse[FIRST], x]], composite[SECOND, inverse[x]]] == True

In[83]:= subclass[fix[composite[inverse[FIRST], x_]], composite[SECOND, inverse[x_]]] := True
```

cod and dom

Lemma.

```
In[15]:= SubstTest[implies, equal[t, y],
            equal[APPLY[y, x], APPLY[t, x]], t → id[fix[y]]] // MapNotNot // Reverse

Out[15]= or[member[x, fix[y]], not[member[x, domain[y]]], not[subclass[y, Id]]] == True

In[16]:= or[member[x_, fix[y_]], not[member[x_, domain[y_]]], not[subclass[y_, Id]]] := True
```

Theorem.

```
In[17]:= Map[implies[and[member[u, y], member[v, z]], #] &,
          SubstTest[or, member[v, fix[t]], not[member[v, domain[t]]], not[subclass[t, Id]],
             t → composite[x, LEFT[u]]] // Reverse // MapNotNot // InvertFix]

Out[17]= or[member[pair[pair[u, v], v], x], not[member[u, y]], not[member[v, z]],
           not[member[pair[u, v], domain[x]]], not[subclass[composite[x, LEFT[u]], Id]]] == True

In[19]:= or[member[pair[pair[u_, v_], v_], x_],
           not[member[pair[u_, v_], domain[x_]]], not[member[u_, y_]],
           not[member[v_, z_]], not[subclass[composite[x_, LEFT[u_]], Id]]] := True
```

Similarly:

```
In[20]:= Map[implies[and[member[u, y], member[v, z]], #] &,
          SubstTest[or, member[u, fix[t]], not[member[u, domain[t]]], not[subclass[t, Id]],
             t → composite[x, RIGHT[v]]] // Reverse // MapNotNot // InvertFix]

Out[20]= or[member[pair[pair[u, v], u], x], not[member[u, y]], not[member[v, z]],
           not[member[pair[u, v], domain[x]]], not[subclass[composite[x, RIGHT[v]], Id]]] == True
```

```
In[22]:= or[member[pair[pair[u_, v_], u_], x_],
           not[member[pair[u_, v_], domain[x_]]], not[member[u_, y_]],
           not[member[v_, z_]], not[subclass[composite[x_, RIGHT[v_]], Id]]] := True
```

Lemma.

```
In[23]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            implies[and[p1, p2, p3], p4], not[implies[p1, p4]], {p1 → member[pair[u, v], dom[x]],
             p2 → member[v, ids[x]], p3 -> subclass[composite[x, RIGHT[v]], Id],
             p4 -> member[pair[pair[u, v], u], x]}]] // Reverse

Out[23]= or[member[pair[pair[u, v], u], x], not[member[u, V]],
           not[member[v, ids[x]]], not[member[pair[u, v], domain[x]]]] == True

In[24]:= (% /. {u → u_, v → v_, x → x_}) /. Equal→ SetDelayed
```

Theorem.

```
In[25]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[u, v],
           implies[member[pair[u, v], t], member[pair[pair[u, v], u], x]], t → dom[x]]]

Out[25]= subclass[dom[x], fix[composite[inverse[FIRST], x]]] == True

In[26]:= subclass[dom[x_], fix[composite[inverse[FIRST], x_]]] := True
```

Corollary.

```
In[27]:= SubstTest[subclass, dom[t], fix[composite[inverse[FIRST], t]], t → flip[x]] // Reverse

Out[27]= subclass[cod[x], inverse[fix[composite[inverse[SECOND], x]]]] == True

In[28]:= subclass[cod[x_], inverse[fix[composite[inverse[SECOND], x_]]]] := True
```

Lemma.

```
In[47]:= Map[not, SubstTest[and, implies[p2, p3],
            implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 → subclass[u, inverse[v]], p2 -> subclass[v, inverse[w]],
             p3 -> subclass[inverse[v], w], p4 → subclass[u, w]}]] // Reverse

Out[47]= or[not[subclass[u, inverse[v]]], not[subclass[v, inverse[w]]], subclass[u, w]] == True

In[48]:= or[not[subclass[u_, inverse[v_]]],
           not[subclass[v_, inverse[w_]]], subclass[u_, w_]] := True
```

Corollary.

```
In[50]:= SubstTest[implies, and[subclass[u, inverse[v]], subclass[v, inverse[w]]],
           subclass[u, w], {u → dom[x], v → inverse[fix[composite[inverse[FIRST], x]]],
            w → inverse[composite[x, inverse[SECOND]]]}] // Reverse

Out[50]= subclass[dom[x], composite[SECOND, inverse[x]]] == True

In[51]:= subclass[dom[x_], composite[SECOND, inverse[x_]]] := True
```

Similarly:

```
In[52]:= SubstTest[subclass, dom[t], composite[SECOND, inverse[t]], t → flip[x]] // Reverse

Out[52]= subclass[cod[x], composite[FIRST, inverse[x]]] == True

In[53]:= subclass[cod[x_], composite[FIRST, inverse[x_]]] := True
```

counterexamples

Each of the inclusions derived in this notebook is in general strict:

dom[x] ⊂ fix[composite[inverse[FIRST], x]] ⊂ domain[x] ⋂ composite[SECOND, inverse[x]]].

The reverse inclusions do not even hold for monoids.

The following inclusion holds:

```
In[79]:= subclass[dom[x], fix[composite[inverse[FIRST], x]]]

Out[79]= True
```

The reverse inclusion need not be true in general, not even for monoids:

```
In[96]:= contains[dom[x], fix[composite[inverse[FIRST], x]]] /. x → NATMUL

Out[96]= False
```

The following inclusion holds:

```
In[84]:= subclass[fix[composite[inverse[FIRST], x]],
           intersection[domain[x], composite[SECOND, inverse[x]]]]

Out[84]= True
```

The reverse inclusion need not be true in general, not even for monoids:

```
In[93]:= contains[fix[composite[inverse[FIRST], x]],
            intersection[domain[x], composite[SECOND, inverse[x]]]] /. x → NATADD

Out[93]= False
```